An Outlier-Resilient Autoencoder for Representing High-Dimensional and Incomplete Data

Di Wu, *Member, IEEE*, Yuanpeng Hu, Kechen Liu, *Student Member, IEEE*, Jing Li, Xianmin Wang, Song Deng, *Member, IEEE*, Nenggan Zheng, *Senior Member, IEEE*, and Xin Luo, *Senior Member, IEEE*

***Abstract*—High-dimensional and incomplete (HDI) data commonly arise in various Big Data-related applications, e.g., recommender systems and bioinformatics. Representation is a learning paradigm to map HDI data into low-dimensional latent space for attracting valuable knowledge and patterns. Currently, deep neural network (DNN) is one of the most popular and successful approaches to represent HDI data due to its powerful nonlinear learning ability. However, previous DNNs-based approaches primarily focused on advancing the sophisticated model structure, neglecting the potential adverse effects of outliers. Unfortunately, outliers usually exist in the collected HDI data. For example, HDI data collected from recommender systems inevitably contain many outlier ratings due to some malicious users. To address this issue, this paper proposes a novel outlier-resilient autoencoder (termed ORA). Its core idea is to design an adaptive Cauchy loss strategy to measure the difference between the observed and predicted data for an autoencoder in representing the HDI data. This strategy leverages a more aggressive Cauchy loss to impose a higher penalty on outlier data with large deviation, while utilizing a smoother Cauchy loss to capture the nuanced, deeper features of HDI data. As such, ORA can dynamically adjust the smoothness of the Cauchy loss during training to handle different levels of data deviation. To evaluate the proposed ORA, extensive experiments are conducted on five benchmark HDI datasets. The results validate that: (1) ORA achieves significantly better representation accuracy than State-of-the-Art DNN- and non-DNN-based models, and (2) ORA possesses higher robustness to outlier data than its peers.**

***Index Terms*—High-dimensional and incomplete data, recommendation model, outlier, cauchy loss, collaborative filtering.**

I. Introduction

MATRICES play a fundamental role in representing pairwise relationships between entities across various application scenarios. They are widely used in multiple domains, including bioinformatics, industrial manufacturing, recommendation systems [1], and internet applications. In the matrix, rows represent users and columns represent items, with each cell recording the outcome or preference of the interaction. The structured nature of these matrices provides a powerful framework for analyzing and modeling interactions between entities, thereby facilitating effective analysis and decision-making in various domains.

However, the high dimensionality and incompleteness(HDI) of matrices [2], [3], [4], [5], [6], [7], [8] pose a central challenge, which is conceptually relevant to real-world application scenarios. For example, in recommendation systems, there are often items that users cannot fully access, leading to incomplete user behavior data. Additionally, data matrices exhibit diversity, influenced by different types of interactions and system factors. Therefore, designing effective methods to represent and mine hidden patterns and knowledge in HDI data matrices is crucial. The data used in recommendation systems [9], [10], [11], [12], [13], [14], [15], [16] are not only high-dimensional and incomplete but also often contain some special outliers. On certain e-commerce platforms and lifestyle service applications, some businesses may artificially inflate their own store reputation through fraudulent transactions and may hire internet trolls to maliciously disparage the products of competitors. These behaviours can produce extreme ratings for certain items, particularly those with competition (e.g., merchandise). These ratings are often unreasonable and can be considered outlier data. Fig. 1 provides an illustration of this issue. There are two common cases in the rating matrix data: normal users, who produce behavioral data that is reasonable and follows a normal logical distribution, and malicious users, who are targeted to produce extreme ratings.

In recent years, the rapid development of deep learning has led to the widespread application of deep neural networks (DNNs) in

Manuscript received 3 May 2024; accepted 7 June 2024. Date of publication 12 August 2024; date of current version 27 March 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62176070, Grant 62272078, Grant 62072127, Grant 62002076, Grant 62293500, Grant 62293501, and Grant 62293505, in part by Chongqing Technical Innovation and Application Development Special Project under Grant CSTB2023TIAD-KPX0037, in part by Chongqing Natural Science Foundation under Grant CSTB2023NSCQ-LZX0069, in part by the Natural Science Foundation of Guangdong Province under Grant 2023A1515011774, and in part by the Science and Technology Program of Guangzhou, China under Grant 202002030131. *(Di Wu and Kechen Liu contributed equally to this work.) (Corresponding authors: Jing Li; Xin Luo.)*

Di Wu and Xin Luo are with the College of Computer and Information Science, Southwest University, Chongqing 400715, China (e-mail: wudi.cigit@gmail.com; luoxin21@gmail.com).

Yuanpeng Hu, Jing Li, and Xianmin Wang are with the School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510002, China (e-mail: yuanpeng@e.gzhu.edu.cn; lijing@gzhu.edu.cn; xianmin@gzhu.edu.cn).

Kechen Liu is with the Department of Computer Science, Columbia University, New York, NY 10027 USA (e-mail: kl3469@columbia.edu).

Song Deng is with the Institute of Advanced Technology, Nanjing University Post & Telecommunication, Nanjing 210003, China (e-mail: zng@cs.zju.edu.cn).

Nenggan Zheng is with the Qiushi Academy for Advanced Studies (QAAS), Zhejiang University Hangzhou, Zhejiang Province 310007, China (e-mail: zng@cs.zju.edu.cn).

The source code of ORA is available at the link: https://github.com/wudi1989/OR_AutoRec.

Recommended for acceptance by B. Xue.

Digital Object Identifier 10.1109/TETCI.2024.3437370

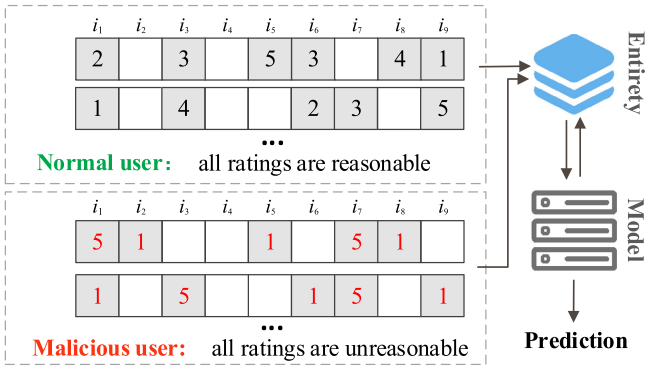

Fig. 1. The difference between normal users and malicious users.

the representation learning of HDI data. Various complex models based on DNNs [17], [18], [19], [20], [21], [22], [23] have been introduced to provide advanced performance in HDI data representation learning. However, most of them focus on improving model performance through complex network architectures while neglecting the negative effects of outliers. To address this issue, we propose a novel outlier-resilient autoencoder (ORA) for representing HDI data. ORA constructs an autoencoder network for deep feature extraction by employing an adaptive Cauchy loss strategy to handle different data characteristics and effectively reduce the sensitivity of the autoencoder to outliers. The main contributions of this paper include:

- It designs an adaptive Cauchy loss strategy for an autoencoder to represent the HDI data. This strategy leverages a more aggressive Cauchy loss to impose a higher penalty on outlier data with large deviation, while utilizing a smoother Cauchy loss to capture the nuanced, deeper features of HDI data.
- It proposes a novel outlier-resilient autoencoder (ORA). It is robust to outliers in representing HDI data by combining the designed adaptive Cauchy loss strategy with an autoencoder.
- It provides the meticulous algorithm design and an in-depth exploration of the proposed ORA model.

Extensive experiments are conducted on five benchmark HDI datasets to evaluate the proposed ORA. By comparing ORA with other advanced models, we validate that ORA possesses higher robustness and accuracy than state-of-the-art DNN- and non-DNN-based models in representing HDI data with outliers. The structure of this paper is as follows: In Section II, we review relevant literature. Section III lays out the required background information. The ORA model is described in detail in Section IV. The experimental results are reported in Sections V, and VI concludes this paper.

II. Related Works

A. DNN-Based Model

The Deep Neural Network (DNN)-based Collaborative Filtering (CF) representation learning model [24], [25], [26] is a widely-used approach for HDI data representation. Compared to traditional CF models [27], [28], [29], [30], it boasts three key advantages: 1) Utilizing nonlinear activation functions, such as Sigmoid [31] and Relu [32], allows for the representation of intricate nonlinear interactions. 2) Its capacity for nonlinear characterisation is greater than that of traditional linear models; and 3) It provides more powerful data mining without the requirement for manual feature extraction.

An autoencoder is a type of deep neural network that is well-suited for both semi-supervised and unsupervised learning. It has been widely used in many autoencoder-based recommendation systems [33]. AutoRec [19] utilizes encoding to extract user/item features and decoding to provide predictions. It applies a reconstruction method that aims to minimize the difference between the input and output. The denoising autoencoder [34] has been presented as a solution to the issue of superfluous information that arises during the encoding process. Strub et al. utilize two Stacked Denoising Autoencoders [35] to process user and item score vectors. These autoencoders understand the underlying user-item relationship and make predictions for missing scores based on implicit representations. Li et al. [36] suggest a method called Marginal Denoising Autoencoder that incorporates many sources of information into the recommendation system.

Some researchers have further improved the model accuracy through constraints on the autoencoder weight matrix, using kernel functions to promote more reasonable weight distributions. The SparseFC [37] model uses a low-dimensional vector parameterization of the weight matrix and employs kernel functions to constrain it through interaction. The Glocal-k model [38] is trained in three stages: first, local kernel functions are used to compute the weight matrix and pre-train the autoencoder; then, the results are merged to obtain a global kernel matrix via kernel processing; finally, the global kernel matrix is refined and fine-tuned using another autoencoder network to produce predictions. Handschutter et al. [39] comprehensive literature review provides an in-depth exploration of the primary models, algorithms, and applications of deep matrix factorization. MIWAE [40], based on importance-weighted autoencoder (IWAE), maximizes a latent tight lower bound of the log-likelihood of observed data, without incurring additional computational overhead due to missing data compared to the original IWAE. Additionally, generative adversarial networks have demonstrated efficacy in modeling complex distributions, with MISGAN [41] serving as an adversarially trained imputer to address missing data.

Deep neural network (DNN) models have shown outstanding abilities in representing the underlying characteristics of Human Development Index (HDI) data. Nevertheless, it is crucial to acknowledge that DNN models sometimes encounter the issue of being sensitive to outliers, which can have a negative impact on performance and resilience of the model.

B. Non-DNN-Based Model

Various non-DNN models, such as latent component analysis models, have been used to express HDI matrices. An often employed method is matrix factorization, which entails breaking down the HDI matrix into two low-rank matrices, often denoting

user factors and item factors. Another approach is the utilization of generalized non-negative matrix factorization, which incorporates non-negativity requirements to more effectively capture the distinctive attributes of HDI matrices. Linked open data-based solutions seek to improve the representation of the HDI matrix by including external data from linked open data sources. Dual regularization is a method that uses two regularization terms to limit the complexity of a model and take into account the connection between latent components and HDI matrices. This leads to more precise representations.

Nevertheless, it is crucial to acknowledge that these studies predominantly concentrate on linear matrix factorization models as a means to tackle the constraints of HDI data. Linear models exhibit reduced susceptibility to outliers, but they may not adequately represent the underlying characteristics of HDI data compared to models based on DNNs.

C. Robust Loss Methods

Many studies have concentrated on enhancing model performance by examining resilient loss functions. For example, previous studies on collaborative filtering have mostly concentrated on the design of interaction encoders, while giving little consideration to loss functions and negative sampling ratios. Mao et al. [42] highlighted the significance of choosing these factors and suggested incorporating cosine contrastive loss into the SimpleX model to improve its resilience. The learning purpose is of paramount importance in recommender systems. The majority of approaches employ pointwise (such as binary cross-entropy) or pairwise (such as BPR) losses, with less emphasis on softmax loss due to its high computational cost. The sampled softmax loss, introduced by [43], is a more efficient alternative to the softmax loss. Conventional wide learning systems exhibit subpar performance in noisy environments because they are highly susceptible to noise. Mlotshwa et al. [44] proposed the Cauchy regularized broad learning system (CRBLS) as a solution to enhance the accuracy of predicting noisy data. The CRBLS incorporates the Cauchy loss function to effectively regulate residuals and effectively handle data that contains noise and outliers. Furthermore, half-quadratic Cauchy non-negative matrix factorization (NMF) and half-quadratic CIM NMF [45] utilize half-quadratic optimization techniques, combined with Cauchy loss functions or conditional independence models, aiming to enhance the fitting capability and robustness to non-negative data with outliers or noise. Guan et al. proposed a technique called Truncated Cauchy NMF [46], which leverages the robust properties of Cauchy loss to robustly learn the feature space on noise-contaminated datasets corrupted by outliers. Note that these robust NMF models were designed to deal with image noise. Nevertheless, the image is full data and therefore does not come within the scope of our study on HDI data representation.

D. Differences in Our Work

In the domain of HDI data representation learning, prior studies have primarily focused on investigating intricate deep neural network models or creating modified versions of autoencoders to improve representation performance. Nevertheless, these endeavors fail to take into account the existence of outliers and their influence on HDI data. Our work specifically aims to enhance the resilience of models to outliers by focusing on the optimization of loss functions and the development of effective loss calculation algorithms. ORA adaptively modifies the γ value in the Cauchy loss function according to the peculiarities of the data distribution. This allows for the identification and penalization of outliers, while also effectively exploring the underlying aspects of HDI data. This resilient loss optimization technique has also demonstrated significant potential in other domains.

TABLE I
DESCRIPTION OF THE SYMBOLS IN THE TEXT

Symbol	Description
U, u	The user set and a user
I, i	The item set and an item
$R, \bar{R}$	The original rating matrix, the predicted rating matrix
$r^{(u)}, r^{(i)}$	Rating vectors for user u and item i
$X^{(i)}, \bar{X}^{(i)}$	The input and output of the AutoEncoder.
r_{ui}	User u corresponds to the rating value of the item i
M	Binary label matrix
m_{ui}	User u corresponds to the tag value of the item i
$\mathcal{F}(\cdot)$	Prediction model
θ	Model parameters
$d(\cdot)$	The error function of the distance between R and $\bar{R}$
$\mathcal{L}(\cdot)$	Loss Calculation Method
$\mathcal{J}(*)$	The empirical risk or the objective function to minimize
E	Error of input and output values
η	Learning rate
λ	Regularization factor
γ	The constants of the Cauchy function
W, V	The weight matrix in the reconstruction function
μ, b	The offset in the refactoring function
f, g	Calculation functions
h^i	Eigenvectors of the latent space
T	Size of the test set
Z	Pre-processed test sample size
rel_i	Represents the relevance of the ith item

III. PRELIMINARIES

This part serves to delineate the symbols employed in the paper and furnish precise explanations for each of them. We provide concise definitions of the rating matrix and rating prediction model for high-dimensional and partial data, together with a thorough discussion of the resilient Cauchy loss.

A. Symbol Appointment

Table I describes in detail the meaning of the symbols covered in this paper and their corresponding descriptions.

B. HDI Data Representation Learning Relevant Definitions

Definition 1 (HDI data matrix): Define the set of users as U and the set of items as I, and $R^{|U|\times|I|}$ denotes a rating matrix of size $|U| \times |I|$, where the rows denote the specified users $u(u \in U)$ and the columns denote the specified items $i(i \in I)$. Each

entry $r_{ui} \in R(u \in U, i \in I)$ of $R^{|U|\times|I|}$ records the user u's interaction data for item i. The higher value denotes the greater interest of a user on a particular item. Each user u is represented by the vector $r^{(u)} = \{R_{u1}, R_{u2}, \ldots, R_{u|I|}\} \in R^{|U|}$, and each item i is represented by the vector $r^{(i)} = \{R_{1i}, R_{2i}, \ldots, R_{|U|i}\} \in R^{|I|}$. In addition, we set a binary token matrix $M^{|U|\times|I|}$ to distinguish between known and unknown user data in $R^{|U|\times|I|}$, where $m_{ui} = 0$ means r_{ui} is unknown and known data is denoted by $m_{ui} = 1$.

Definition 2 (Outliers): In statistics, an outlier is defined by considering the divergence of a measurement from the mean of the data collection, as well as the standard deviation. To quantify the distance between each measurement in a collection of measurements N and the mean of the data set, one can calculate the mean $\bar{n}$ and standard deviation σ. An outlier is defined as a measurement that deviates from the mean by more than two times the standard deviation (i.e., $|n_i - \bar{n}| > 2\sigma$). A measurement is classified as a highly severe outlier when the difference between the measurement and the mean is greater than three times the standard deviation (i.e., $|n_i - \bar{n}| > 3\sigma$). These standards are utilised to detect observations in the dataset that differ significantly from the average, enabling researchers to discover and address outliers in order to enhance the precision of data analysis and model predictions.

Definition 3 (Rating prediction model): The missing entry prediction-based representation method uses observed HDI data to train a parametric model $f(\theta)$ for imputing the missing entries. We use

$$\mathcal{F}\left(R^{|U|\times|I|}, \theta\right) \to \bar{R}^{|U|\times|I|} \tag{1}$$

to represent the representation process, where θ denotes the hyperparameters of the model. The model is trained and continuously optimized to reduce the error between the prediction result and the ground truth, which can be expressed as follows.

$$\mathcal{L}(f) = \min \sum_{u\in U, i\in I} M \otimes d\left(r_{ui} - \mathcal{P}\left(r_{ui}, \theta\right)\right) \tag{2}$$

Where $d(\cdot)$ is the error function used to calculate the error function of the predicted data and the input data. $\mathcal{P}(\cdot)$ represents the output of the representation model. $\otimes$ denotes the Hadamard product (the component-wise multiplication), and M is a $|U| \times |I|$ binary token matrix:

$$M_{u,i} = \begin{cases} 1, & \text{if } R_{u,i} \text{ is known.} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

According to the PAC learning theory [47], [48], as long as the training sample is large enough, the learned model can achieve a very low true risk, which means that the performance of the model will be approximately optimal.

C. Cauchy Loss

As shown in Fig. 2(a), there is a noticeable discrepancy among different loss functions. The horizontal axis, Error, represents the deviation between predicted and true values, while the vertical axis shows the corresponding loss. As the error increases, the loss for the L_2-norm and L_1-norm rises significantly more than that for the Cauchy loss function. This indicates that other loss functions focus more on large errors during training, resulting in greater sensitivity to such data and potentially affecting model performance. The non-smoothness of the L_1 loss function can also lead to instability. In contrast, the low sensitivity to outliers of robust Cauchy loss function makes it an ideal choice for optimizing the model. We use

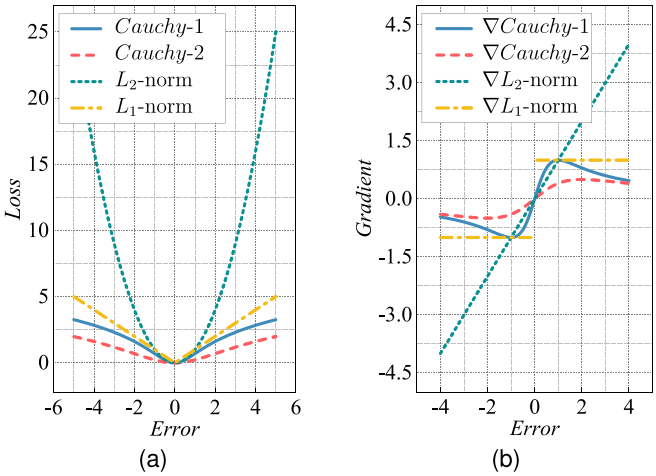

Fig. 2. Comparison of the L_1-norm, L_2-norm, Cauchy-1 ($\gamma = 2$) and Cauchy-1 ($\gamma = 4$). (a) Trend of each loss function. (b) Trend of the gradients of each loss function.

$$g(x) = \ln\left(1 + \left(\frac{x}{\gamma}\right)^2\right) \tag{4}$$

to denote the Cauchy loss function. where x denotes the error and γ is the sole parameter of the Cauchy loss function.

The representation model uses gradient descent to generate results. Each weight W in the representation model network is updated through backpropagation, which can be expressed as (taking W_1 as an example):

$$x = \mathrm{y} - h(\theta) \tag{5}$$

$$L_k = g(x) = g\left(\mathrm{y} - h(\theta)\right) \tag{6}$$

where y represents the value in the input vector, $h(\theta)$ represents the output value generated by the neural network, and L_k represents the loss of $k$nodes in the output layer. The model calculates the gradient of the weight parameter $w_1$by the chain derivative rule. The calculation formula is as follows:

$$\frac{\partial L_k}{\partial w_1} = \frac{\partial L_k}{\partial h(\theta)} \times \frac{\partial h(\theta)}{\partial net_k} \times \frac{\partial net_k}{\partial w_1} \tag{7}$$

substitutes the loss function $g(x)$ into the equation (8) to calculate.

$$\frac{\partial L_k}{\partial w_1} = \frac{\partial g(x)}{\partial x} \times \frac{\partial x}{\partial h(\theta)} \times \frac{\partial h(\theta)}{\partial net_k} \times \frac{\partial net_k}{\partial w_1} \tag{8}$$

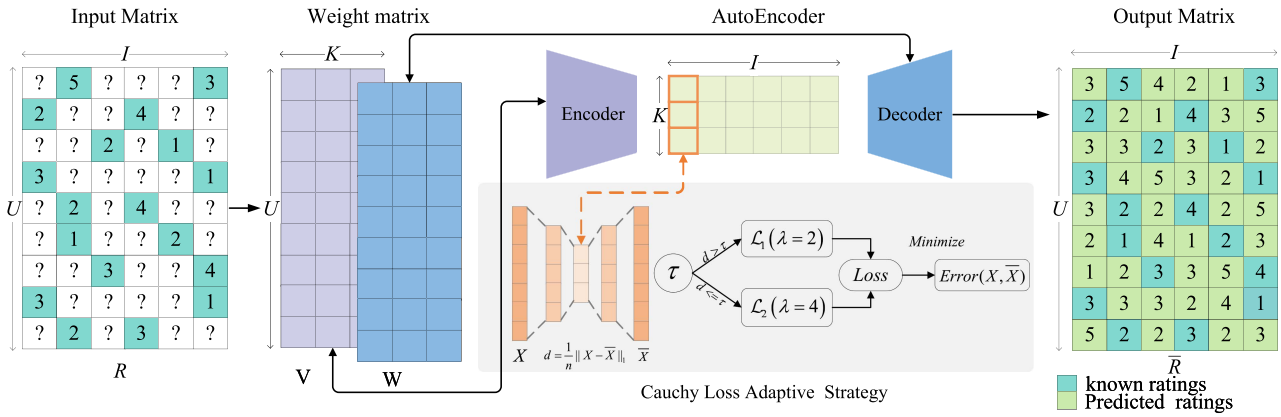

Fig. 3. ORA architecture for representing HDI data.

Combined with the learning rate of the model, the gradient update equation of W_1 is as follows:

$$w_1^+ = w_1 - \eta \times \frac{\partial L_k}{\partial w_1} \tag{9}$$

As shown in Fig. 2(b), the gradient descent process calculates the partial derivatives of each parameter, including the loss function. The figure shows how the gradient of the loss function changes with increasing error. The derivative of L_1 loss function remains constant, while the derivative of L_2 loss function increases linearly. The gradient of the Cauchy loss function increases nonlinearly and steadily, allowing for stable multivariate gradient updates and reducing sensitivity to large errors.

IV. Model Description

In this section, we provide an overview of the ORA model, including its overall architecture, a detailed introduction to the autoencoder, a theoretical analysis of the adaptive Cauchy loss strategy, the complete model training optimization process, and the algorithm implementation steps.

A. Structure of the ORA

The structure of the ORA is illustrated in Fig. 3. We utilize a rating prediction task to thoroughly evaluate the capacity of ORA to precisely depict HDI data. ORA employs an autoencoder consisting of three fully connected layers, which include an encoder, a hidden layer, and a decoder. The encoder functions as the initial layer for the original data, the hidden layer represents the underlying feature space of the input layer, and the decoder is responsible for reconstructing the output of the encoder. The ORA model is trained using the HDI data matrix as input; its architecture is defined by the weight matrices V and W of the network. The weight matrices in the fully connected layers establish connections between the input layer and the hidden and decoder layers, facilitating the mapping of inputs to outputs. The weight matrices execute linear transformations on the input data and integrate them with bias vectors to reconstitute the input data. During the encoding phase, the model employs a nonlinear activation function to compute the dot product of the input matrix R and the parameter matrix V, resulting in the creation of the hidden layer matrix H. The decoding phase involves obtaining the reconstructed output matrix $\bar{R}$ by performing a dot product operation between the hidden layer matrix H and the parameter matrix W.

At a more granular level, we utilize the Cauchy loss adaptive approach to compute the loss of the model and optimize its parameters. The Cauchy loss function with more aggressive characteristics intensifies the penalty for samples with substantial deviations, while the Cauchy loss function with smoother features captures more profound aspects of HDI data. This combination strikes a balance between how the loss function handles extreme values and its ability to capture the overall structure of the data. By dynamically choosing the Cauchy loss function, the model effectively harnesses the benefits of several loss functions at different stages of training, enabling more flexible adaptation to various data distributions.

B. Autoencoder

Autoencoder, as an unsupervised generative model, is utilized for learning effective data representations. We employed a three-layer fully connected architecture, comprising an input layer, a hidden layer (encoding layer), and an output layer (decoding layer), along with weight matrices W and V connecting the neurons across layers. The input layer is responsible for receiving raw data and encoding it into a low-dimensional representation (hidden layer) in the latent space, followed by decoding back to the original data space (output layer). The specific computational process is outlined as follows. First is the encoding process:

$$H = f(VX + \mu) \tag{10}$$

where X is the input data, V is the weight matrix connecting the input layer and the hidden layer; f and μ respectively represent the activation function and the bias vector of the hidden layer; H denotes the hidden layer, serving as the low-dimensional representation of the input data in the latent space. Next is the decoding process:

$$\bar{X} = g(WH + b) \tag{11}$$

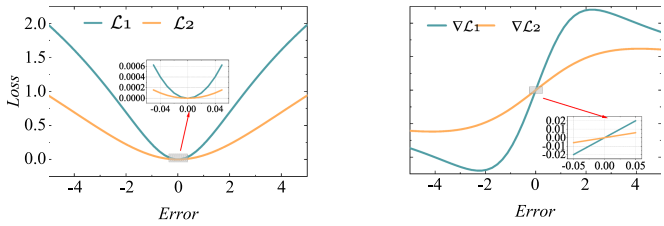

Fig. 4. Comparing Cauchy losses under different scale factors. (a) Trend of the $\mathcal{L}_1$ and $\mathcal{L}_2$. (b) Trend of the gradients of the $\nabla\mathcal{L}_1$ and $\nabla\mathcal{L}_2$.

where W is the weight matrix connecting the hidden layer and the output layer; g and b respectively denote the activation function and the bias vector of the output layer; $\bar{X}$ represents the output of the decoder, which signifies the reconstructed data obtained through decoding the hidden layer H.

The objective of the AutoEncoder is to minimize the reconstruction error between the input X and the output $\bar{X}$.

$$\mathcal{L} = \sum_{X^{(i)} \in R} \left\| \left(X^{(i)} - \bar{X}^{(i)} \right) \odot m^{(i)} \right\|^2 \tag{12}$$

Incorporating a regularization term to prevent model overfitting and enhance model generalization, the complete objective function of the AutoEncoder is as follows:

$$\mathcal{J}(*) = \sum_{X^{(i)} \in R} \left\| \left(X^{(i)} - \bar{X}^{(i)} \right) \odot m^{(i)} \right\|^2 + \frac{\lambda}{2} \left(\|W\|_2^2 + \|V\|_2^2 \right) \tag{13}$$

where $m_i \in M$ is a binary indicator matrix used to record the positions of known data in the input.

C. Adaptive Cauchy Loss Strategy

We have devised a Cauchy loss adaptive strategy for computing the reconstruction error between the input X and the output $\bar{X}$, aiming to reduce the sensitivity of the AutoEncoder to outliers. The smoothness of the Cauchy loss is determined by the scale factor γ. We have employed two variations of the Cauchy loss, denoted as $\mathcal{L}_1$ and $\mathcal{L}_2$.

$$\mathcal{L}_1 = \ln\left(1 + \left(\frac{X - \bar{X}}{2}\right)^2\right) \tag{14}$$

$$\mathcal{L}_2 = \ln\left(1 + \left(\frac{X - \bar{X}}{4}\right)^2\right) \tag{15}$$

The $\mathcal{L}_1$ loss has a more pronounced distribution, whereas $\mathcal{L}_2$ has a smoother profile. The two losses and their gradients within a limited region display a notable disparity, as illustrated in Fig. 4(a) and (b).

In the adaptive strategy, we adopt a more aggressive $\mathcal{L}_1$ approach to enhance the penalty on outlier samples exhibiting significant deviations, thereby reducing the model sensitivity to outliers. Simultaneously, we utilize a smoother $\mathcal{L}_2$ to capture intricate representations in HDI data, thereby enhancing the model representational capacity. The outlined adaptive strategy is as follows: We set an adaptive threshold based on statistical analysis. Firstly, we compute the average of all known ratings in the training set. Next, we determine the threshold by calculating the product of the mean and the corresponding confidence interval at a 1% confidence level using statistical analysis.

$$\tau = \frac{1}{n} \sum_{i=1}^{n} r_i \times z_{0.01} \tag{16}$$

The ORA computes the mean squared error between the input vector X and output vector $\bar{X}$ at each training iteration after determining the threshold. If the error exceeds the threshold τ, the model utilizes the $\mathcal{L}_1$ loss; otherwise, it employs the $\mathcal{L}_2$ loss. The computations are as stated below:

$$\mathcal{L}_{ORA} = \begin{cases} \mathcal{L}_1, & \text{if } ||X - \bar{X}|| > \tau \\ \mathcal{L}_2, & \text{otherwise} \end{cases} \tag{17}$$

We conducted a theoretical analysis of the Cauchy adaptive strategy, exploring how it penalizes outlier samples. The equation for computing the Cauchy loss is as follows:

$$\mathcal{L}(\cdot) = \ln\left(1 + \left(\frac{X - \bar{X}}{\gamma}\right)^2\right) \tag{18}$$

The parameter γ serves as a scale factor; larger values of γ result in smoother Cauchy loss, whereas smaller values render the loss function more aggressive. As $\bar{X}$ deviates from X, the norm $||X - \bar{X}||$ increases. In such scenarios, we adaptively adjust the loss by using a threshold; when $||X - \bar{X}|| > \tau$, we opt for a smaller value of γ. This increases the computed result of $(X - \bar{X}/\gamma)^2$, consequently elevating the overall loss.

After the loss has been computed, the model conducts backpropagation and gradient descent through an optimizer to update the network parameters. The equation for computing the gradient of the Cauchy loss is as follows:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{2(X - \bar{X})}{\gamma^2 \left(1 + \left\| \frac{X - \bar{X}}{\gamma} \right\|^2\right)} \cdot \frac{\partial}{\partial W}(X - \bar{X}) \tag{19}$$

When γ becomes smaller, the square of the γ term in the denominator decreases, leading to an increase in the overall gradient magnitude. Larger gradients imply larger parameter update magnitudes, accelerating the model convergence speed and enabling it to adapt more quickly to training data. In such scenarios, the model is more likely to escape local optima and converge to the global optimum faster. Larger gradients typically indicate that the loss function is more sensitive to parameter changes. When dealing with outliers, larger gradients allow the model to adjust to outliers more quickly, thereby reducing the impact of outliers on model parameters.

D. Modeling Process of ORA

Integrating Cauchy Loss Adaptive Strategy into AutoEncoder for ORA Model Training. The objective of ORA is to minimize the error between actual and predicted values to generate a reconstructed matrix that closely resembles the input matrix.

The objective function is as follows:

$$\mathcal{J}(*) = \sum_{X^{(i)} \in R} \ln\left(1 + \left(\frac{(X^{(i)} - \bar{X}^{(i)}) \odot m^{(i)}}{\gamma}\right)^2\right) + \frac{\lambda}{2}\left(\|W\|_2^2 + \|V\|_2^2\right) \tag{20}$$

Where according to the Cauchy loss adaptive strategy, the value of $\gamma = \|X^{(i)} - \bar{X}^{(i)}\| > \tau \ ? \ 2 : 4$; for the reconstructed output $\bar{X}$, its calculation equation is as follows:

$$\bar{X}^{(i)} = g\left(W \times f\left(V \times X^{(i)} + \mu\right) + b\right) \tag{21}$$

ORA employs backpropagation and gradient descent to update the parameter matrices W and V in the AutoEncoder. The Adam algorithm combines the advantages of Adagrad (an adaptive learning rate gradient descent algorithm) and the momentum gradient descent algorithm. This method is highly efficient in terms of computational resource utilization, consumes minimal memory, and is well-suited for tackling complex data and large-scale parameter optimization problems. Therefore, we adopt the Adam optimization algorithm to optimize the model. The equation for updating parameters is as follows:

$$\begin{aligned} k_t &\leftarrow \beta_1 k_{t-1} + (1-\beta_1)\nabla f_t \\ v_t &\leftarrow \beta_2 v_{t-1} + (1-\beta_2)\nabla f_t \\ \bar{k}_t &\leftarrow k_t / (1-\beta_1^t) \\ \bar{v}_t &\leftarrow v_t / (1-\beta_2^t) \\ \theta_t &\leftarrow \theta_{t-1} - \eta \cdot \bar{k}_t / (\sqrt{\bar{v}_t} + \varepsilon) \end{aligned} \tag{22}$$

The variables k_t, v_t, ∇f_t, and θ_t represent the first momentum, second momentum, gradient, and target parameters, respectively, at step t. The symbols β_1 and β_2 represent the decay rate, while the symbol η represents the learning rate. The ε is a minimum number that is set to ensure that the denominator does not equal zero.

The entire modeling process of ORA is obtained through the encoding and decoding steps of the AutoEncoder. The complete overview of this technique is outlined as follows:

To start, initialize the necessary parameter values for the model, including the two weight matrices V and M, the bias vectors μ and b, the learning rate η, and the regularization coefficient λ. We input the complete input matrix R into the model. During training, R is divided into n vectors X as inputs.

$$\begin{aligned} V^{k\times n} &= \left\{v^{(1)}, v^{(2)}, \ldots, v^{(n)}\right\} \\ W^{m\times n} &= \left\{w^{(1)}, w^{(2)}, \ldots w^{(n)}\right\} \\ R^{m\times n} &= \left\{X^{(1)}, X^{(2)}, \ldots, X^{(n)}\right\} \end{aligned} \tag{23}$$

Pass an input vector to the encoder to encode and extract the m-dimensional input vector into a k-dimensional feature vector, where k is equal to 500. The encoding procedure of the i-th input vector is illustrated by the (24), (25). The variables f and μ indicate the activation function and offset, respectively.

$$h^{(i)} \leftarrow Encoder\left(v^{(i)}, X^{(i)}, \mu\right) \tag{24}$$

$$\begin{cases} h_1^{(i)} = f\left(X_1^{(i)} \times v_1^{(i)} + X_2^{(i)} \times v_1^{(i)} + \ldots + X_m^{(i)} \times v_1^{(i)} + \mu\right) \\ h_2^{(i)} = f\left(X_1^{(i)} \times v_2^{(i)} + X_2^{(i)} \times v_2^{(i)} + \ldots + X_m^{(i)} \times v_2^{(i)} + \mu\right) \\ \ldots \\ h_k^{(i)} = f\left(X_1^{(i)} \times v_k^{(i)} + X_2^{(i)} \times v_k^{(i)} + \ldots + X_m^{(i)} \times v_k^{(i)} + \mu\right) \end{cases} \tag{25}$$

Subsequently, the feature vector is decoded by decoding the output vector corresponding to the i-th position. The activation function and offset are denoted by g and b, respectively.

$$\bar{X}^{i} \leftarrow Decoder\left(h^{(i)}, w^{(i)}, b\right) \tag{26}$$

$$\begin{cases} \bar{X}_1^{(i)} = g\left(h_1^{(i)} \times w_1^{(i)} + h_2^{(i)} \times w_1^{(i)} + \ldots + h_k^{(i)} \times w_1^{(i)} + b\right) \\ \bar{X}_2^{(i)} = g\left(h_1^{(i)} \times w_2^{(i)} + h_2^{(i)} \times w_{12}^{(i)} + \ldots + h_k^{(i)} \times w_2^{(i)} + b\right) \\ \ldots \\ \bar{X}_m^{(i)} = g\left(h_1^{(i)} \times w_m^{(i)} + h_2^{(i)} \times w_m^{(i)} + \ldots + h_k^{(i)} \times w_m^{(i)} + b\right) \end{cases} \tag{27}$$

After obtaining the reconstructed output $\bar{X}$, ORA iteratively trains and adjusts the weight parameters W and V using (20) to minimize the objective function.

E. Algorithm Analysis and Design

Algorithm 1 provides a clear and concise description of the sequential actions involved in the ORA paradigm. The model necessitates the rating matrix R, the number of iterations E, the learning rate η, the regularization coefficient λ, and the constant γ in the Cauchy function as inputs. Before training, we establish the weight matrices W and V as well as the offsets μ and b with random values. In order to accurately obtain the recorded rating values from the given input data, we create a marker matrix M following the instructions provided in equation (3). Before conducting the training, it is necessary to compute the threshold τ using the known ratings in the training set. In the first layer, we employ a Sigmoid activation function to compute the feature space $h^{(i)}$. In the second layer, we utilize an Identity activation function to generate the output layer $\bar{X}^{(i)}$. The loss is determined by applying equation (20). Ultimately, the complete model optimizes and updates the model parameters W and V by employing the Adam optimizer along with backpropagation and gradient descent.

The time complexity of ORA is analyzed. The derived equation is as follows:

$$\begin{aligned} C = O(|U| \times |I|) + O(|U| \times K) + O(K \times |I|) \\ + O\left(|I| \times |U| \times E\right) \approx O(|I| \times |U| \times E) \end{aligned} \tag{28}$$

The computational cost of the proposed technique is denoted as C, with K representing the number of hidden layer units in the autoencoder neural network. It is assumed that K is significantly smaller than U and I. The primary computational expense is incurred during steps 4 to 12 in Algorithm 1. This involves multiplying the size of the input matrix R, which has dimensions $(|U| \times |I|)$, with the number of training iterations E.

Algorithm 1: ORA Algorithm.

Input: $R, E, \eta, \lambda, \gamma$
Output: $\bar{R}$

Initializing W, V, μ, b randomly
Build the binary tag matrix M
Calculate static threshold τ
while *not at end of this document* **do**
  for $i = 1, 2, 3, 4, ..., I$ **do**
    $h^{(i)} = f\ (\ V \times X^{(i)} + \mu)$
    $\bar{X}^{(i)} = g\ (\ W \times h^{(i)} + b)$
    $\mathcal{L}_{ORA} = ||X^{(i)} - \bar{X}^{(i)}|| > \tau\ ?\ \mathcal{L}_1 : \mathcal{L}_2$
    $\mathcal{J}(\mathcal{L}_{ORA}, \theta)$
  end
  update W, V with Adam($\mathcal{J}(\mathcal{L}_{ORA}, \theta)$, η)
end
return

TABLE II
BRIEF INFORMATION ABOUT ALL DATASETS

No.	Name	Users	Items	Ratings	Density*	Max/Min value
D1	MovieLens100K	943	1682	100000	6.30%	5/1
D2	MovieLens1M	6040	3952	1000209	4.19%	5/1
D3	Yahoo	15400	1000	365704	2.37%	5/1
D4	HetRec	2113	10109	855598	4.01%	5/0.5
D5	Protein-protein interaction	181	181	021783	5.85%	1/0.15

*Density denotes the percentage of observed entries in the user-item matrix.

V. EXPERIMENTS

In future trials, our objective is to investigate the following research inquiries:

- RQ.1. Does the ORA model effectively represent both the original HDI matrices in comparison to other models?
- RQ.2. Has the implementation of the Cauchy Loss Adaptive Strategy resulted in a beneficial effect on ORA?
- RQ.3. What is the effect of outliers on the predicted accuracy of the ORA model?

A. General Settings

Datasets: For our experimental research, we utilize five reputable HDI datasets that encompass diverse fields including movie ratings, e-commerce, and bioinformatics. The datasets consist of MovieLens100 K, MovieLens1M, Yahoo, HetRec, and the protein-protein association dataset 1_zyr_224308. The table labeled "Table II" presents a thorough data profile for the study, including detailed information on the dimensions, sparsity, and rating scale of the datasets. The selection of these datasets is meticulously done to guarantee comprehensive representation across all domains. In order to ensure that our experiments are consistent and reliable, we continuously divide the data into training and testing sets using an 80%–20% split.

Evaluation Metrics: The primary objective of representation learning on HDI data is to forecast the absent values. In order to evaluate the precision of the model depiction, we employ the root mean square error (RMSE) and mean absolute error (MAE) metrics. The calculations are performed in the following manner:

$$MAE = \left(\sum_{r_{ui} \in T} |\bar{r}_{ui} - r_{ui}| \right) / |T|$$

$$RMSE = \sqrt{\left(\sum\nolimits_{r_{ui} \in T} (\bar{r}_{ui} - r_{ui})^2 \right) / |T|}$$

Furthermore, we utilize the *NDCG@k* and *Hit@k* metrics, which are widely employed in recommender systems, to assess the effectiveness of our proposed model in ranking prediction experiments. In the ranking studies, the test dataset underwent preprocessing by classifying samples into positive (ratings 4 and 5) and negative (ratings 1, 2, and 3) categories. A test case was created by pairing positive samples with 100 randomly selected negative samples from the same user. If the number of negative samples available was less than 100, all of them were included. Each sample in the test case was assigned prediction scores to determine the relative ordering of positive samples. The ranking was used to calculate the NDCG@k and Hit@k scores. The calculations are performed in the following manner:

$$NDCG@k = \frac{1}{Z} \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

$$Hit@k = \frac{\textit{Number of relevant items in top-k}}{Z}$$

Baselines: We evaluate the performance of ORA by comparing it to eight other models: AutoRec, NRR, MF, GLocal-K,

TABLE III
DESCRIPTIONS OF ALL THE CONTRASTING MODELS

Model	Description
AutoRec [19]	AutoRec is a collaborative filtering technique that utilizes autoencoder technology to effectively represent intricate data patterns and connections.
NRR [49]	NRR is a versatile deep learning model capable of performing multiple tasks, including the learning of HDI matrix representations.
MF [50]	MF is a latent factor model that utilizes matrix decomposition techniques and incorporates collaborative filtering features.
GLocal-K [38]	GLocal-K utilizes a highly optimized global kernel modeling deep learning algorithm to depict HDI data in a lower-dimensional space.
SparseFC [37]	SparseFC modifies the weight matrix by utilizing low-dimensional vectors that interact via kernel functions.
FML [51]	FML utilizes metric learning as a latent factor model, integrating collaborative filtering within the distance space.
MetaMF [52]	MetaMF employs a multilayer perceptron architecture to effectively train representations of high-dimensional input, which makes it particularly suitable for tasks involving rating prediction.
ADNLF [53]	A representation model based on non-negative latent factors (NLF). This model constructs a generalized objective function using $\alpha - \beta-$divergence to enhance its capability in representing various HDI data.

TABLE IV
RATING PREDICTION PERFORMANCE COMPARISON OF DIFFERENT MODELS ON STANDARD DATA SETS

Dataset	Metric	AutoRec	NRR	MF	Glocal-K	SparseFC	FML	MetaMF	ADNLF	ORA(Ours)
Movielen100K	RMSE	0.8883 •	0.9258 •	0.9016 •	0.8905 •	0.8909 •	0.8973 •	0.9149 •	0.8933 •	0.8821
	MAE	0.6966 •	0.7269 •	0.7077 •	0.6999 •	0.7035 •	0.7094 •	0.7152 •	0.7041 •	0.6906
Movielens1M	RMSE	0.8491 •	0.8758 •	0.8510 •	0.8291	0.8349	0.8457 •	0.8719 •	0.8477 •	0.8375
	MAE	0.6697 •	0.6874 •	0.6696 •	0.6483	0.6517	0.6632 •	0.6856 •	0.6678 •	0.6567
Yahoo	RMSE	1.1730 •	1.2249 •	1.1891 •	1.1930 •	1.1695 •	1.1624	1.2504 •	1.1722 •	1.1634
	MAE	0.8807 •	0.9535 •	0.9346 •	0.8913 •	0.8860 •	0.9235 •	0.9562 •	0.8892 •	0.8794
Hetrec	RMSE	0.7480 •	0.7725 •	0.7524 •	0.7553 •	0.7465 •	0.7496 •	0.7685 •	0.7501 •	0.7396
	MAE	0.5697 •	0.5844 •	0.5688 •	0.5888 •	0.5770 •	0.5693 •	0.5799 •	0.5654 •	0.5607
1_zyr_224308	RMSE	0.1593 •	0.1361 •	0.1552 •	0.1662 •	0.1682 •	0.1093	0.1751 •	0.1439 •	0.1154
	MAE	0.1092 •	0.0864 •	0.1073 •	0.1173 •	0.1154 •	0.0732	0.1233 •	0.0967 •	0.0762
Statistical analysis	Win/Loss	10/0	10/0	10/0	8/2	8/2	7/3	10/0	10/0	**73/7***
	F-rank	4.4	7.5	5.8	5.1	4.1	3.8	8.3	4.3	**1.7**
	p-value	0.00097	0.00097	0.00097	0.00781	0.00781	0.01367	0.00097	0.00097	–

• The cases that ORA wins the comparison. * The total Win/Loss cases of ORA.

SparseFC, FML, MetaMF and ADNLF. Table III provides a succinct summary of various models.

Implementations Details: The hyperparameters and network layers of all models were adjusted to optimize performance. The final testing result is generated by the optimal model, which achieves the minimum prediction error in the validation set during the training process. The model training process will end when it reaches the predetermined number of training iterations. All the tests are performed on a GPU server equipped with two 2.4 GHz Xeon Gold 6240R processors, each with 24 cores, 376.40 GB of RAM, and a single Tesla V100 GPU. In order to facilitate the replication of our research, we will make our source code publicly available after the anonymous review process is complete.

B. Performance Comparison (RQ.1)

1) Comparison of Rating Prediction Performance: The experiment assessed the performance of several models on five benchmark data sets. In order to achieve a fair comparison, the parameters of each model were meticulously tuned to get their optimal performance. The empirical findings are displayed in Table IV. In order to assess the statistical significance of the variations in performance across the models, we performed both a Friedman test and a Wilcoxon signed-rank test on the experimental data. The Friedman test was employed to assess the overall performance of different models on multiple data sets by calculating the F-rank value. A lower F-rank score signifies superior forecast accuracy. Conversely, the Wilcoxon signed-rank test is a statistical technique that does not rely on specific assumptions about the data distribution. It is used to compare the performance of two models that are paired together. The statistical significance threshold, as indicated by the *p*-value, was employed to ascertain if the prediction accuracy of ORA was significantly superior than that of the comparison models. According to the Table IV, Our observations are as follows:

- ORA consistently demonstrates the lowest Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) in the majority of comparisons. The win/loss ratio is 73/7.
- ORA has the lowest F-rank, indicating its superior overall performance.
- All of the *p*-values obtained from the Wilcoxon signed-rank test are less than the significance level of 0.05. This suggests that the prediction accuracy of ORA is superior to that of other competitors from a different viewpoint.

2) Comparison of Personalized Ranking Performance: To further showcase the efficacy of the ORA model in personalized ranking, we assessed its performance using ranking assessment measures including NDCG@k and Hit@k. We conducted an analysis of its effectiveness across various suggestion list lengths (k) and performed a significance test to confirm the observed variations. The test results are displayed in Table V. Based on the table, the following may be observed:

- ORA demonstrates superior performance compared to other baseline models in the majority of scenarios, with an impressive Win/Loss ratio of 155/5 overall.
- ORA demonstrates superior overall performance, achieving the lowest F-Rank number.
- All p-values obtained from the ORA analysis are less than 0.05, which suggests that there are statistically significant differences when comparing the test results of different models.

The research findings clearly demonstrate that the ORA model is highly effective in predicting ratings and creating individualized rankings, outperforming other similar methods. As a result, the performance of ORA shows significant potential for a wide range of practical applications.

C. Comparison Test of Adaptive Cauchy Loss (RQ.2)

In order to assess the beneficial effects of the Cauchy loss adaptation approach on the ORA model, we carried out three sets of comparative experiments (ORA-1, ORA-2, and ORA). ORA-1 utilized a more assertive Cauchy loss function with a constant parameter $\gamma = 2$, ORA-2 employed a gentler Cauchy loss function with a constant value $\gamma = 4$, and ORA utilized a Cauchy adaptive method. The figures depicting the results of the comparison tests for Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are labeled as Figs. 5 and 6.

Upon analyzing the comparison results depicted in the figures, we have made the following observations:

TABLE V
PERSONALIZED RANKING PERFORMANCE COMPARISON OF DIFFERENT MODELS ON STANDARD DATA SETS

DataSet	Metric	AutoRec	NRR	MF	GLocal-K	SparseFC	FML	MetaMF	ADNLF	ORA(Ours)
MovieLens100K	Hit@5	0.7638	0.7254 ●	0.7402 ●	0.7579 ●	0.7544 ●	0.7493 ●	0.6971 ●	0.7555 ●	0.7631
	Hit@10	0.8963 ●	0.8781 ●	0.8818 ●	0.8964 ●	0.8895 ●	0.8918 ●	0.8533 ●	0.8849 ●	0.8997
	NDCG@5	0.5754 ●	0.5354 ●	0.5509 ●	0.5734 ●	0.5587 ●	0.5643 ●	0.5014 ●	0.5552 ●	0.5789
	NDCG@10	0.6186 ●	0.5849 ●	0.5972 ●	0.6187 ●	0.6027 ●	0.6106 ●	0.5524 ●	0.5992 ●	0.6228
Movielens1M	Hit@5	0.6893 ●	0.6545 ●	0.6861 ●	0.7013	0.7011	0.6887 ●	0.6169 ●	0.6916 ●	0.6989
	Hit@10	0.8313 ●	0.8054 ●	0.8268 ●	0.8327 ●	0.8293 ●	0.8292 ●	0.7762 ●	0.8322 ●	0.8367
	NDCG@5	0.5169 ●	0.4805 ●	0.5141 ●	0.5299	0.5295	0.5188 ●	0.4436 ●	0.5198 ●	0.5286
	NDCG@10	0.5631 ●	0.5296 ●	0.5598 ●	0.5713 ●	0.5705 ●	0.5644 ●	0.4956 ●	0.5668 ●	0.5734
Yahoo	Hit@5	0.5462 ●	0.4985 ●	0.5611 ●	0.5472 ●	0.5462 ●	0.5731 ●	0.5044 ●	0.5635 ●	0.5881
	Hit@10	0.8089 ●	0.7582 ●	0.7881 ●	0.8001 ●	0.8179 ●	0.8059 ●	0.7701 ●	0.7952 ●	0.8268
	NDCG@5	0.3509 ●	0.3193 ●	0.3622 ●	0.3682 ●	0.3467 ●	0.3884 ●	0.3032 ●	0.3541 ●	0.3904
	NDCG@10	0.4348 ●	0.4031 ●	0.4354 ●	0.4491 ●	0.4535 ●	0.4335 ●	0.3884 ●	0.4324 ●	0.4679
Hetrec	Hit@5	0.5602 ●	0.5083 ●	0.5473 ●	0.5301 ●	0.5572 ●	0.5478 ●	0.4657 ●	0.5587 ●	0.5637
	Hit@10	0.7091 ●	0.6739 ●	0.7008 ●	0.6879 ●	0.7069 ●	0.7026 ●	0.6358 ●	0.7067 ●	0.7135
	NDCG@5	0.4137 ●	0.3653 ●	0.3994 ●	0.3831 ●	0.4122 ●	0.4009 ●	0.3247 ●	0.4094 ●	0.4169
	NDCG@10	0.4619 ●	0.4189 ●	0.4491 ●	0.4343 ●	0.4607 ●	0.4511 ●	0.3796 ●	0.4625 ●	0.4655
1_zyr_224308	Hit@5	0.3064 ●	0.4501●	0.2292 ●	0.2473 ●	0.2543 ●	0.7512 ●	0.7323 ●	0.5186 ●	0.8079
	Hit@10	0.4831 ●	0.6032 ●	0.3952 ●	0.2474 ●	0.4572 ●	0.8574 ●	0.8428 ●	0.6542 ●	0.8871
	NDCG@5	0.2141 ●	0.3293 ●	0.1454 ●	0.4431 ●	0.1955 ●	0.6201 ●	0.6211 ●	0.4047 ●	0.6847
	NDCG@10	0.2624 ●	0.3783 ●	0.1984 ●	0.2372 ●	0.2414 ●	0.6611 ●	0.6533 ●	0.4486 ●	0.7106
Statistical analysis	Win/Loss	19/1	20/0	20/0	18/2	18/2	20/0	20/0	20/0	**155/5***
	F-rank	4.225	7.55	6.7	4.45	4.625	4.15	7.65	4.4	**1.25**
	p-values	0.00006	0.00005	0.00005	0.00009	0.00008	0.00005	0.00005	0.00005	–

● The cases that ORA wins the comparison. * The total Win/Loss cases of ORA.

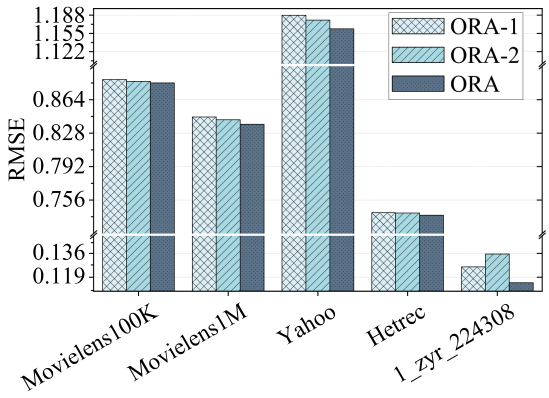

Fig. 5. Adaptive Cauchy loss comparison results (RMSE).

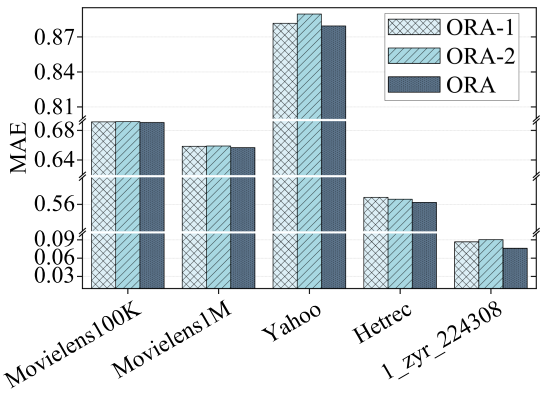

Fig. 6. Adaptive Cauchy loss comparison results (MAE).

- ORA consistently achieves superior performance compared to the other two models using fixed Cauchy loss techniques on five typical datasets. The exceptional result demonstrates the efficacy of the adaptive Cauchy loss technique.
- The experiments demonstrate that ORA-1 and ORA-2, while utilizing distinct Cauchy loss algorithms, display varying levels of performance across different datasets. The varying distributional properties of the datasets have a certain impact on the judgment made by the Cauchy loss function.

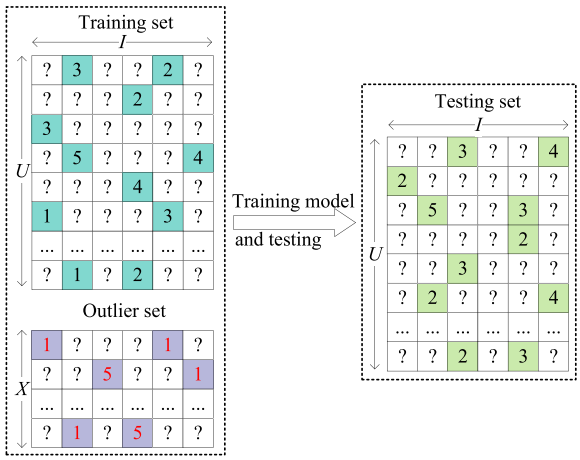

Fig. 7. An example to add outliers.

Overall, these observations emphasize the improved adaptability of the adaptive Cauchy loss technique when dealing with data that has varying distribution characteristics. The strategy effectively adapts by penalizing outlier samples with substantial deviations and capturing intricate latent features in HDI data. It utilizes distinct Cauchy loss methods at different phases of training to maximize their respective strengths. This strategy

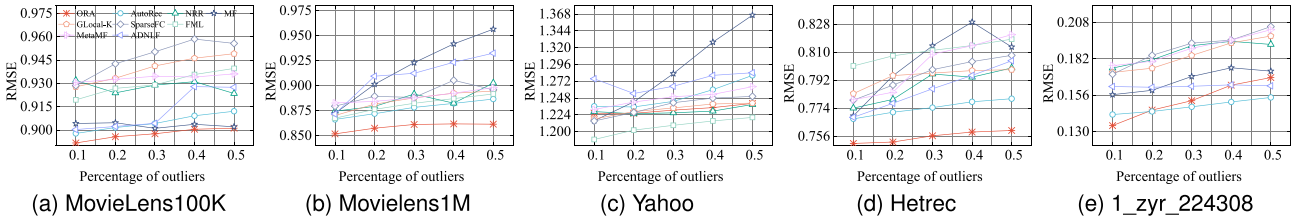

Fig. 8. Results of the root mean square error (RMSE) evaluation of outlier data sensitivity tests for ORA and comparison models, where $\lambda = 1, \eta = 0.001$.

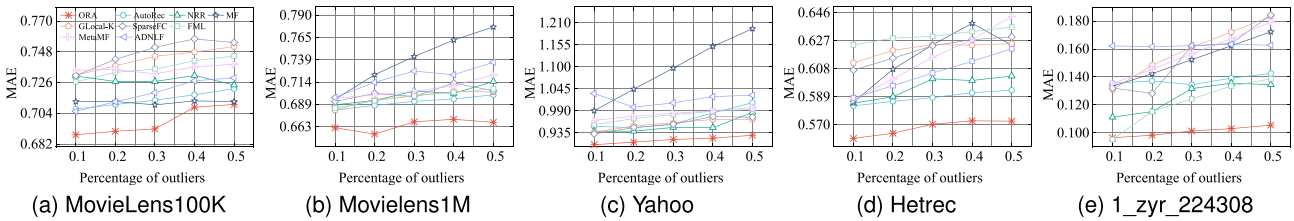

Fig. 9. Results of the mean absolute error (MAE) evaluation of outlier data sensitivity tests for ORA and comparison models, where $\lambda = 1, \eta = 0.001$.

enhances the performance and accuracy of the model across various types of data.

D. Outlier Data Sensitivity Tests (RQ.3)

Table II demonstrates that the often employed datasets for HDI representation learning are frequently characterized by a large number of dimensions and a scarcity of data points. In order to demonstrate the sensitivity to outliers, we conducted a simulation where we intentionally introduced malicious user infiltration to artificially raise the proportion of outlier data. The methodology for including outliers is outlined as follows: (1) Integrate different amounts of abnormal data into the original dataset. (2) Every row of outlier data contains both 1 and 5 categories of outliers, with each category representing 50% of the row. (3) The percentage of outlier entities added varies from 10% to 50% of the known entities, increasing by 10% each time. (4) Only outlier entities are included in the training set. Fig. 7 depicts a particular instance.

The findings of the outlier sensitivity experiment, as shown in Fig. 8 and Fig. 9, demonstrate that the ORA model consistently outperforms alternative models at different degrees of outlier density. Although there has been an increase in the density of outliers, the ORA model consistently performs well, while the performance of other models varies depending on the extent to which outliers affect the data. Furthermore, the ORA model demonstrates exceptional performance on datasets with high levels of sparsity, such as MovieLens100 K, Movielens1M, and Hetrec.

Based on the comprehensive sensitivity test results, the following observations can be made:

- As anomalous data is gradually supplied, the RMSE and MAE values either increase consistently or vary. The presence of outliers has a significant impact on the prediction accuracy, as indicated by this pattern. The model performance is negatively affected by the existence of outliers, particularly when they are more prominent. This is because the model struggles to make consistent predictions when unexpected or extreme data are encountered.
- As the proportion of outliers increases, the ORA model consistently maintains stable prediction accuracy, demonstrating its lower sensitivity to outliers compared to other DNN and non-DNN models. This reduced sensitivity enhances the model adaptability to outliers and boosts its predictive performance, resulting in greater robustness and reliability when managing outlier data.

The results of the outlier sensitivity test showcase the robustness of the ORA model, which effectively mitigates the adverse effects of outlier data. These findings emphasize the resilience of ORA against outliers and confirm its superiority over other models in this regard.

VI. Conclusion

This study proposes an outlier-resilient autoencoder (termed ORA) to accurately represent the high-dimensional and incomplete (HDI) data. Conventional representation learning models based on deep neural networks (DNNs) depend on intricate parameter adjustments to enhance the performance of the model. Nevertheless, ORA emphasizes the detrimental effect of excessive behavioral data on predictive accuracy. To tackle this problem, this study suggests enhancing the loss function of the model by integrating more resilient loss functions and appropriate loss calculation algorithms. This would decrease the model susceptibility to extreme interaction data (outliers) and enhance its resilience. Comparative research reveals that the adaptive Cauchy loss technique is superior to the commonly employed L_1 and L_2 loss functions in classic DNN models when it comes to handling outliers. The experimental results on five benchmark datasets demonstrate that the proposed model has reduced susceptibility to outliers, increased robustness, and enhanced prediction accuracy. In addition, our analysis demonstrates that

ORA possesses a certain degree of resilience against attacks initiated by rogue individuals. In the future, we intend to extend the utilization of ORA.

REFERENCES

[1] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, 2019.

[2] J. Chen, Y. Yuan, T. Ruan, J. Chen, and X. Luo, "Hyper-parameter-evolutionary latent factor analysis for high-dimensional and sparse data from recommender systems," *Neurocomputing*, vol. 421, pp. 316–328, 2021.

[3] S. J. Choudhury and N. R. Pal, "Deep and structure-preserving autoencoders for clustering data with missing information," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 5, no. 4, pp. 639–650, Aug. 2021.

[4] J. Chen, R. Wang, D. Wu, and X. Luo, "A differential evolution-enhanced position-transitional approach to latent factor analysis," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 2, pp. 389–401, Apr. 2023.

[5] Z. Sun, S. Kong, and W. Wang, "Sparse learning of higher-order statistics for communications and sensing," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 1, pp. 13–22, Feb. 2020.

[6] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, "A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4425–4445, May 2023.

[7] X. Luo, Y. Zhou, Z. Liu, and M. Zhou, "Fast and accurate non-negative latent factor analysis of high-dimensional and sparse matrices in recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3897–3911, Aug. 2023.

[8] Y. Yuan, Q. He, X. Luo, and M. Shang, "A multilayered-and-randomized latent factor model for high-dimensional and sparse matrices," *IEEE Trans. Big Data*, vol. 8, no. 3, pp. 784–794, Jun. 2022.

[9] D. Liu et al., "Mitigating confounding bias in recommendation via information bottleneck," in *Proc. 15th ACM Conf. Recommender Syst.*, 2021, pp. 351–360, doi: 10.1145/3460231.3474263.

[10] D. Wu, P. Zhang, Y. He, and X. Luo, "A double-space and double-norm ensembled latent factor model for highly accurate web service QoS prediction," *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 802–814, Mar./Apr. 2023.

[11] Y. Zheng, C. Gao, X. Li, X. He, Y. Li, and D. Jin, "Disentangling user interest and conformity for recommendation with causal embedding," in *Proc. WWW '21: Web Conf. 2021*, Ljubljana, Slovenia, 2021, pp. 2980–2991, doi: 10.1145/3442381.34497883.

[12] Z. Ovaisi, K. Vasilaky, and E. Zheleva, "Propensity-independent bias recovery in offline learning-to-rank systems," in *Proc. SIGIR '21: 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Canada, 2021, pp. 1763–1767, doi: 10.1145/3404835.3463097.

[13] X. Wang, R. Zhang, Y. Sun, and J. Qi, "Combating selection biases in recommender systems with a few unbiased ratings," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, 2021, pp. 427–435, doi: 10.1145/3437963.3441799.

[14] D. Wu, Y. He, X. Luo, and M. Zhou, "A latent factor analysis-based approach to online sparse streaming feature selection," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 11, pp. 6744–6758, Nov. 2022.

[15] Q. Guo et al., "A survey on knowledge graph-based recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3549–3568, Aug. 2022.

[16] Y. Yin, Z. Cao, Y. Xu, H. Gao, R. Li, and Z. Mai, "Qos prediction for service recommendation with features learning in mobile edge computing environment," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1136–1145, Dec. 2020.

[17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182, doi: 10.1145/3038912.3052569.

[18] M. Fu, H. Qu, Z. Yi, L. Lu, and Y. Liu, "A novel deep learning-based collaborative filtering model for recommendation system," *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 1084–1096, Mar. 2019.

[19] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 111–112, doi: 10.1145/2740908.2742726.

[20] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proc. Ninth ACM Int. Conf. Web Sea. Data Mining*, 2016, pp. 153–162, doi: 10.1145/2835776.2835837.

[21] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proc. Int. Joint Conf. Artif. Intell.*, Melbourne, Australia, 2017, vol. 17, pp. 3203–3209.

[22] M. Wang, Y. Lin, G. Lin, K. Yang, and X.-m. Wu, "M2grl: A multi-task multi-view graph representation learning framework for web-scale recommender systems," in *Proc. KDD '20: Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. & Data Mining*, 2020, pp. 2349–2358, doi: 10.1145/3394486.3403284.

[23] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. World Wide Web Conf.*, 2018, pp. 689–698, doi: 10.1145/3178876.3186150.

[24] M. F. Aljunid and M. Doddaghatta Huchaiah, "Multi-model deep learning approach for collaborative filtering recommendation system," *CAAI Trans. Intell. Technol.*, vol. 5, no. 4, pp. 268–275, 2020.

[25] R. Srinivas, N. Verma, E. Kraka, and E. C. Larson, "Deep learning-based ligand design using shared latent implicit fingerprints from collaborative filtering," *J. Chem. Inf. Model.*, vol. 61, no. 5, pp. 2159–2174, 2021.

[26] G. B. Martins, J. P. Papa, and H. Adeli, "Deep learning techniques for recommender systems based on collaborative filtering," *Expert Syst.*, vol. 37, no. 6, 2020, Art. no. e12647.

[27] D. Wu, M. Shang, X. Luo, and Z. Wang, "An l_1-and-l_2-norm-oriented latent factor model for recommender systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5775–5788, Oct. 2022.

[28] C.-D. Wang, Z.-H. Deng, J.-H. Lai, and S. Y. Philip, "Serendipitous recommendation in e-commerce using innovator-based collaborative filtering," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2678–2692, Jul. 2019.

[29] Y. Zhang, K. Meng, W. Kong, and Z. Y. Dong, "Collaborative filtering-based electricity plan recommender system," *IEEE Trans. Ind. Informat.*, vol. 15, no. 3, pp. 1393–1404, Mar. 2019.

[30] V. W. Anelli, A. Bellogín, T. Di Noia, and C. Pomo, "Reenvisioning the comparison between neural collaborative filtering and matrix factorization," in *Proc. Fifteenth ACM Conf. Recommender Syst.*, 2021, pp. 521–529, doi: 10.1145/3460231.3475944.

[31] W. Liu, Z. Wang, Y. Yuan, N. Zeng, K. Hone, and X. Liu, "A novel sigmoid-function-based adaptive weighted particle swarm optimizer," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 1085–1093, Feb. 2021.

[32] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with ReLU activation function," *Ann. Statist.*, vol. 48, no. 4, pp. 1875–1897, 2020.

[33] T. Liang, M. Chen, Y. Yin, L. Zhou, and H. Ying, "Recurrent neural network based collaborative filtering for QoS prediction in IoV," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2400–2410, Mar. 2022.

[34] Y. Jhamb, T. Ebesu, and Y. Fang, "Attentive contextual denoising autoencoder for recommendation," in *Proc. ACM SIGIR Int. Conf. Theory Inf. Retrieval*, 2018, pp. 27–34, doi: 10.1145/3234944.3234956.

[35] T. Dai, L. Zhu, Y. Wang, and K. M. Carley, "Attentive stacked denoising autoencoder with Bi-LSTM for personalized context-aware citation recommendation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 553–568, 2020.

[36] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 811–820, doi: 10.1145/2806416.2806527.

[37] L. K. Müller, J. N. P. Martel, and G. Indiveri, "Kernelized synaptic weight matrices," in *Proc. 35th Int. Conf. Mach. Learn.*,Stockholmsmässan, Stockholm, Sweden, 2018, pp. 3651–3660.

[38] S. C. Han, T. Lim, S. Long, B. Burgstaller, and J. Poon, "Glocal-k: Global and local kernels for recommender systems," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 3063–3067, doi: 10.1145/3459637.3482112.

[39] P. De Handschutter, N. Gillis, and X. Siebert, "A survey on deep matrix factorizations," *Comput. Sci. Rev.*, vol. 42, 2021, Art. no. 100423.

[40] P.-A. Mattei and J. Frellsen, "Miwae: Deep generative modelling and imputation of incomplete data sets," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4413–4423.

[41] S. C.-X. Li, B. Jiang, and B. Marlin, "Misgan: Learning from incomplete data with generative adversarial networks," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–20. [Online]. Available: https://openreview.net/forum?id=S1lDV3RcKm

[42] A. Rusiecki, "Trimmed robust loss function for training deep neural networks with label noise," in *Proc. Artif. Intell. Soft Computing: 18th Int. Conf.*, Zakopane, Poland, 2019, pp. 215–222.

[43] J. Wu, X. Wang, X. Gao, J. Chen, H. Fu, and T. Qiu, "On the effectiveness of sampled softmax loss for item recommendation," *ACM Trans. Inf. Syst.*, vol. 42, no. 4, pp. 1–26, 2024.
[44] T. Mlotshwa, H. van Deventer, and A. S. Bosman, "Cauchy loss function: Robustness under Gaussian and Cauchy noise," in *Proc. Southern Afr. Conf. Artif. Intell. Res.*, 2022, pp. 123–138.
[45] B.-W. Chen and Y.-H. Wu, "Partially observed visual IoT data reconstruction based on robust half-quadratic collaborative filtering," *IEEE Internet Things J.*, vol. 11, no. 3, pp. 3690–3701, Feb. 2024.
[46] N. Guan, T. Liu, Y. Zhang, D. Tao, and L. S. Davis, "Truncated Cauchy non-negative matrix factorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 246–259, Jan. 2019.
[47] A. Burrello, A. Marchioni, D. Brunelli, S. Benatti, M. Mangia, and L. Benini, "Embedded streaming principal components analysis for network load reduction in structural health monitoring," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4433–4447, Mar. 2021.
[48] L. Zhao, X. Zhao, H. Zhou, X. Wang, and X. Xing, "Prediction model for daily reference crop evapotranspiration based on hybrid algorithm and principal components analysis in Southwest China," *Comput. Electron. Agriculture*, vol. 190, 2021, Art. no. 106424.
[49] P. Li, Z. Wang, Z. Ren, L. Bing, and W. Lam, "Neural rating regression with abstractive tips generation for recommendation," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 345–354, doi: 10.1145/3077136.3080822.
[50] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[51] S. Zhang, L. Yao, B. Wu, X. Xu, X. Zhang, and L. Zhu, "Unraveling metric vector spaces with factorization for recommendation," *IEEE Trans. Ind. Inform.*, vol. 16, no. 2, pp. 732–742, Feb. 2020.
[52] Y. Lin et al., "Meta matrix factorization for federated rating predictions," in *Proc. 43 rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 981–990.
[53] Y. Yuan, R. Wang, G. Yuan, and L. Xin, "An adaptive divergence-based non-negative latent factor model," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 53, no. 10, pp. 6475–6487, Oct. 2023.

Di Wu (Member, IEEE) received the Ph.D. degree from the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China, in 2019. He is currently a Professor with the College of Computer and Information Science, Southwest University, Chongqing. He has more than 80 publications, including 23 IEEE Transactions papers and several conference papers on AAAI, ICDM, WWW, and IJCAI. His research interests include machine learning and data mining. He is an Associate Editor for *Neurocomputing* (IF 6) and *Frontiers in Neurorobotics* (IF 3.1). For more information please visit his homepage: https://wudi1989.github.io/Homepage/

Yuanpeng Hu received the B.S. degree in computer science and technology from the College of Computer Science and Big Data Science, Jiujiang University, JiangXi, China, in 2021. He is currently working toward the M.S. degree in computer technology with Guangzhou University, Guangzhou, China. His research interests include data mining and machine learning.

Kechen Liu (Student Member, IEEE) is currently working toward the Bachelor of Science degree in computer science with Columbia University, New York, NY, USA. She is also a Research Assistant with the Department of Computer Science, Columbia University. Her research interests include data science and artificial intelligence.

Jing Li received the B.S. degree from Inner Mongol Normal University, China, in 2010, the M.S. degree from Shaanxi Normal University, Xi'an, China, in 2013, and the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China. She is currently with Guangzhou University, Guangzhou, China. Her research interests include cloud computing, applied cryptography, and privacy-preserving.

Xianmin Wang received the B.S. degree from Suzhou University, Jiangsu, China, in 2006, the M.S. degree in computer science from the Jiangxi University of Science and Technology, Jiangxi, China, in 2013, and the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2017. He is currently with the Institution of School of Computer Science, Guangzhou University, Guangzhou, China. His research interests include deep learning and image processing and understanding.

Song Deng (Member, IEEE) received the Ph.D. degree in information network from the Nanjing University of Posts and Telecommunication, Nanjing, China, in 2009. From 2009 to 2012, he was a Research Fellow with the State Grid Electric Power Research Institute, Nanjing. From 2012 to 2014, he was a Research Fellow with the China Electric Power Research Institute, Beijing, China. He is currently a Full Professor with the Nanjing University of Posts and Telecommunication. Up to now, he has authored or coauthored more than 50 papers in international journals, including IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, and *ACM Transactions on Internet Technology*. His research interests include data security, information security of cyber-physical systems, data mining, and knowledge engineering.

Nenggan Zheng (Senior Member, IEEE) received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 2002 and 2009, respectively. He is currently a Full Professor of computer science with Zhejiang University. His research interests include cyborg robotics, artificial intelligence, and embedded systems.

Xin Luo (Senior Member, IEEE) received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from the Beihang University, Beijing, China, in 2011. He is currently a Professor of data science and computational intelligence with the College of Computer and Information Science, Southwest University, Chongqing, China. He has authored or coauthored more than 200 papers (including more than 140 IEEE Transactions papers) in his research field, which interestsBig Data analysis and intelligent control. Dr. Luo was the recipient of the Outstanding Associate Editor Award from IEEE/CAA Journal of Automatica Sinica in 2020, and from IEEE Transactions on Neural Networks and Learning Systems in 2022. He is currently an Associate Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and IEEE/CAA JOURNAL OF AUTOMATICA SINICA. His page is https://scholar.google.com/citations?user=hyGlDs4AAAA&hl=zh-TW.